(12) UK Patent Application (19) GB (11) 2 331 814 (13) A

(43) Date of A Publication 02.06.1999

(21) Application No 9724364.6

(22) Date of Filing 19.11.1997

(71) Applicant(s)
International Business Machines Corporation
(Incorporated in USA - New York)
Armonk, New York 10504, United States of America

(72) Inventor(s)
Andrew Liam Massey
Sohail Syyed

(74) Agent and/or Address for Service
D P Litherland
IBM United Kingdom Limited, Intellectual Property
Department, Mail Point 110, Hursley Park,
WINCHESTER, Hampshire, SO21 2JN,
United Kingdom

(51) INT CL^6
G06F 9/445

(52) UK CL (Edition Q)
G4A AFL

(56) Documents Cited
None

(58) Field of Search
UK CL (Edition P) G4A AFL
INT CL^6 G06F
ONLINE:WPI

(54) Abstract Title
Pre-emptive download of software in data processing network

(57) Software modules (e.g. dynamically loadable code such as java applets) are pre-emptively transferred in a data processing network from a host data processing system for execution on a client data processing system. The software modules are of a number of different types, each type providing a different service at the client system (e.g. e-mail, text editor etc). The host system determines the type of a first software module transferred to a client and makes a prediction, based on this determination, as to the type of software module that may next be required at the client. The predicted software module is then transferred to the client system in advance of any request for same from the client.

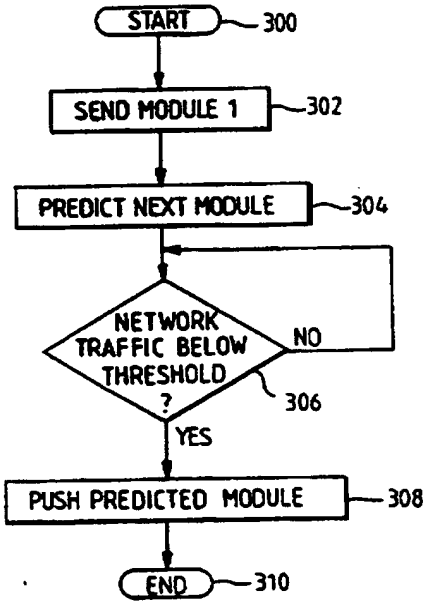

FIG. 4

GB 2 331 814 A

2331814

PRE-EMPTIVE DOWNLOAD OF SOFTWARE IN DATA PROCESSING NETWORK

Technical Field of the Invention

The present invention relates to the pre-emptive transfer of software from a host system to target systems in a data processing network.

Background of the Invention

In a typical networking environment, multiple client data processing systems are connected to one or more server systems. In a first common arrangement, each client system includes an operating system and optionally other software, stored on a hard file within the client. Other application software e.g. word processing, database software etc, held on storage associated with the server system, is accessed as needed by the client system.

In an alternative network arrangement, the client system may have little or no mass storage capability, in which case the operating system and other software required by the user is downloaded from the server when needed by the client system and stored in volatile memory.

In existing networks, software and other data is generally transferred from the server at the request of the client system; either by specific action on the part of the client user or automatically during execution of a process on the client. When the server system is busy, requests for software may await their turn in a queue, thereby resulting in significant delays. Moreover, even if there is no queue, and the request for software is actioned immediately on receipt at the server, there is nevertheless a loss of time occasioned by the need for the requesting client to transmit the request to the server together with a loss of time involved in the production of the requested software and the communication of the requested software over the connecting link to the client. At times of high network traffic, the total delay incurred can be significant and may adversely impact the activities of the user of the client system.

US 5029104 describes a technique whereby the host system anticipates the need at the connected workstation for certain types of data object. The host includes heuristic logic circuits which respond to

data objects (e.g. electronic mail of interest to the workstation user) received at the host to determine the likelihood of use of each object by a workstation. Data objects likely to be used at the workstation are voluntarily transmitted by the host to the work station prior to a request being made by the workstation.

Thus in this prior art, the host system pre-emptively sends data objects to the workstations on a determination that they may be of use or interest to the workstation user. This determination is based on the receipt of data objects at the host system and does not take account of the activity of the user at the workstation.

Disclosure of the Invention

According to a first aspect of the invention there is provided a method for pre-emptively transferring software modules in a data processing network from a host data processing system for execution on a target data processing system, wherein there are a plurality of types of software modules, each type providing a different service at the target system, the method comprising: determining, at the host system, the type of a first software module transferred to a target system; predicting, based on the determination of the type of the first module, a second software module type that may next be required at the target system; and automatically transferring a module of the second type to the target system.

According to a second aspect of the present invention there is provided a server data processing system comprising: means for communicating with a plurality of client data processing systems; means for transferring a plurality of types of software modules to the plurality of client systems for execution thereon, each type of software module providing a different service at a target system; means for monitoring the types of software modules sent to the target systems; and means, responsive to a determination that a first type of software module has been communicated to a target system, for predicting that a software module of a second type will be subsequently required at the target system, and for causing the transferring means to pre-emptively transfer a software module of the second type to the target system.

Thus in accordance with the present invention, there is provided a method and system for making a prediction, based on the type of software

module which has been transferred to the client system, as to the type of software module which will next be required by the client. In a preferred arrangement, the prediction is based on the history of software module usage associated with each client, which history is compiled by the server over a period of time.

In a preferred embodiment of the invention, the level of traffic on the network is monitored and the predicted module is automatically transferred only when the measured traffic is below a threshold value.

In one preferred arrangement, the software modules are dynamically loadable (e.g. Java applets) for execution via a java-enabled browser or java virtual machine on the client system.

According to a further aspect of the invention, there is provided a computer program product comprising: a computer usable medium having computer readable program code means embodied in said medium for pre-emptively transferring software modules in a data processing network from a host data processing system for execution on a target data processing system, wherein there are a plurality of types of software modules, each type providing a different service at the target system, the computer readable program code means comprising: computer readable program code means for determining, at the host system, the type of a first software module transferred to a target system; computer readable program code means for predicting, based on the determination of the type of the first module, a second software module type that may next be required at the target system; and computer readable program code means for automatically transferring a module of the second type to the target system.

Preferred embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings.

Brief Description of the Drawings

Figure 1 is a schematic of a data processing network comprising a host system connected to a plurality of client data processing systems;

Figure 2 is a schematic illustration of the major components of the client and host systems of Figure 1;

Figure 3 is a schematic illustration of the interaction between the elements of the host system used in the preferred embodiment of the present invention;

Figure 4 is a flowchart showing the steps of one preferred embodiment of the present invention; and

Figure 5 is a schematic illustration of an alternative network configuration with which the present invention may be employed.

Detailed Description of the Invention

Referring firstly to Figure 1, there is shown, in schematic form, a local area network (LAN) 10 in which a preferred embodiment of the invention is implemented. The network of Figure 1 is shown configured as an Ethernet LAN but may alternatively be configured according to a Token-Ring or any other suitable topology. As will be described below, the invention is also useful in the context of the internet or intranet network. In Figure 1, the network comprises a server computer system 20 (which in the present embodiment may be an IBM PC 730 computer system connected for communication by a link 26 with a plurality of client computer systems 30, 32, 34, 36. The client computer systems may be personal computers based on the Intel X86 family of microprocessors or other forms of computer system. As will be described below, the client may alternatively be a 'network computer' (for example the IBM Network Station) including a Java Virtual Machine or a Java-enabled browser. Each client system includes a LAN adapter card or network interface card (NIC) 38, 40, 42, 44 to provide communication with the server computer over link 26.

Control of resources on the network including communication between server and clients is effected by means of a network operating system (NOS) e.g. Netware from Novell Inc having a 'server' component which executes on the main processor(s) of the server computer system and a corresponding 'requester' component which executes on the main processor of each client computer system. Other suitable network operating systems include OS/2 LAN Server and OS/2 WARP Server, both from IBM Corporation. In the present embodiment, the Internetwork Packet Exchange (IPX) communication protocol is employed in relation to the Netware operating system. In other embodiments, TCP/IP may be the protocol of choice.

Figure 2 is a simplified block diagram showing the connection of server computer system 20 to client system 30 over communication link 26. The client system includes a keyboard 131 and a display 132 operating under the control of control logic in the form of main CPU 133 which is connected by a system bus to system memory (RAM) 134 and non-volatile memory (ROM) 135, in which is stored system BIOS including POST code. The client system further includes a network adapter card 137 which, in the present embodiment is an ethernet card but in an alternative embodiment may be a token-ring adapter card. This adapter card includes non-volatile memory in the form of ROM in which is stored code employed in providing communication between the client and server.

It will be noted that the client system of the present embodiment is not provided with mass storage device(s) e.g. in the form of a magnetic disk drive (hard file). Furthermore, if it is desired to prevent the client user from introducing software or data into the client system, the client system is advantageously not provided with a diskette drive, CD-ROM drive or similar.

The server computer system of Figure 2 includes a keyboard 121 attached to a system unit 122 including a main CPU 123, system RAM 124, system ROM 125 and mass storage capability 126, typically in the form of multiple magnetic disk drives constituted in a RAID (redundant array of independent disks) arrangement. The server system may optionally include a display 127 (if the network administrator requires direct interaction with the server system) and other storage devices such as a diskette drive (not shown) and CD-ROM drive 129. Communication over the link 26 is provided by input/output logic 128 which may take the form of an adapter card.

Stored on the disk drives of the server RAID is a variety of different software including operating system software and a variety of different types of software modules for downloading to, and use by, the client systems. These software modules are relatively small pieces of code which are designed to provide a different service at the client system. For example, there may be module types designed to provide e-mail, text editing, calendar, spreadsheet and other services to the client user. The modules are transferred to the client systems either at the specific request of a user (person or process) or in a pre-emptive manner in accordance with the invention.

The software modules may advantageously be designed to be dynamically loadable on the client system in which case when the module is transferred to the client, it automatically loads and is presented to the client user e.g. in a window. The module may further be designed to run once, after which it is discarded. Alternatively, it can be cached for later use by the client. One example of a suitable software module is the Java Applet (Java is a trade mark of Sun Microsystems, Inc). In this case, the client system would comprise either a Java-enabled browser (e.g. Netscape Navigator available from Netscape Communications) or a Java Virtual Machine (JVM) running on a suitable operating system such as Windows NT.

With reference to Figure 3, there is shown the interrelationship of a group of software components 200 (Netware modules in the present embodiment) which execute on the server system to provide the functionality of the preferred embodiment of the invention. LAN Analyzer software 210 (e.g. LANAnalyzer available from Novell, Inc) monitors the traffic on the network in a conventional manner in order to provide information to the controller 205 as to the level of traffic on the LAN. There are many suitable LAN analysers on the market, for example LANAnalyzer available from Novell, Inc)

Data Gatherer software 220 is designed to run as a background task to build up a history of the usage of various software modules on various client systems. This history is compiled by the Data Gatherer task over a period of days or weeks to obtain a model of the modules requested by a particular client, the time of day they are requested and also the typical sequence in which the different module types are used.

For example, client X may invariably request a calendar software module at the start of the business day in order to check the times of existing scheduled meetings and/or to add further meeting information. This particular user may then invariably request a spreadsheet module in order to update financial or other information relevant to the previous business day. Another client Y may first request an e-mail module followed by a calendar module. It will be appreciated that the usage history is likely to vary between clients depending on the roles of the individual users within the organisation where the LAN is located.

This history information is compiled by the data gatherer software and is used in the present invention to make a prediction as to the

future requirements of any of the client systems. Thus, when client system X requests a calendar module, this is monitored by the data gatherer task. The controller software includes a predictive algorithm 225 which predicts, based on the information from the data gatherer, which type of software module is likely to be used next by the client system X. In the present case, the history indicates that the client X invariably requests the spreadsheet module after the calendar module. In this case, the controller causes the push task 230 to push the spreadsheet module to client X in advance of any request from the user of client X. In a preferred arrangement, the transfer of the next predicted module is conditional on a determination (based on the traffic information from the LAN Analyzer task) that the LAN traffic is below a predefined or calculated threshold. This arrangement allows for improved balancing of the load on the network.

In an alternative arrangement, there is no data gatherer software for building up a history of the module usage by various clients. Instead, the prediction is based on a listing of associations between the types of software module. For example, if a client first requests a text editor then it might be reasonable to expect that the next module required will be an e-mail communication module. This latter module can then be pre-emptively transferred to the client in advance of the client request for that module. The association of modules can be stored in any suitable form e.g. a table, in the server system.

Irrespective of the mode of prediction, the predicted software modules will ideally be transferred just prior to the predicted use by the client in which case the module is dynamically loaded by the client such that, for example a window representing the service pops up on the screen of the client. The ability to achieve this ideal arrangement will depend on the sophistication of the data gatherer software; the available bandwidth on the network at any one time and also on the predictability of the client user. The dynamically loadable module can either be discarded after use, in which case the module would have to be resent by the server, or alternatively can be cached in anticipation of a subsequent invocation.

The module need not be dynamically loadable but may rather be transferred to cache memory at the client for subsequent invocation by the user. Any suitable arrangement may be used to control the caching of the software modules in the client. For example, when the available cache

is full, the software modules can be purged on a FIFO basis or alternatively each module may have associated with it a priority (e.g. text editor more important than spreadsheet) and space can be freed in the cache according to the module priority.

With reference now to Figure 4, there is shown a flowchart of the steps involved in a preferred method of the present invention. The method starts as indicated at 300. The server system sends, at step 302, a first software module type to one of the attached client systems. This may be either at the request of client or alternatively in a pre-emptive fashion by the server. The Control software makes a prediction at step 304, based on the history information compiled by the Data Gatherer, as to the type of software module which will, in all likelihood be required by the same client. At step 306, the Control software makes a determination as to whether the LAN traffic is below a threshold value and if so causes the push software, at step 308, to push the predicted module to the client in question. The process ends at step 310.

As has already been indicated, the usage history associated with a particular client will be specific to the user of that client. The success of the predictive algorithm will thus depend on the client system being used by the same user. This will generally be the case even in an open office environment including a group of workstations. Individuals in such an office will typically be allocated a particular system on which to work. Thus if user A uses workstation WS10 every Monday for example, the data gatherer software will note that the pattern in software module use remains relatively consistent from Monday to Monday. Thus the server system will predictively supply the appropriate software module to workstation WS1 in advance of the arrival of user A on a Monday morning. Equally, the server will predictively supply further software modules, prior to any request from workstation WS1 during the remainder of the day.

If for some reason user B chooses to use workstation WS10 on a Monday morning and makes a request for a software module which is normally not required by WS10, this is noted by the Data Gatherer software. Based on this, instead of predictively supplying the normal 'Monday' sequence of software modules, the Control software may either cease making any prediction or alternatively may predict software use purely on the association of software module types in the manner described above.

The present invention may also be used in the context of the internet or intranet. Figure 5 is a schematic illustration of such an arrangement including a host system 400 connected via the internet or intranet 410 to a plurality of client systems, three of which are shown as systems 412, 414, 416. The host system may have the same configuration as the host of Figure 1. The client systems will include either a Java-enabled browser 420 or a Java Virtual Machine for executing Java applet software modules received from the host system. Each client optionally includes a cache 422 for temporary storage of the software modules. The host system is enabled to compile and maintain a history of module usage with reference to the individual IP addresses of the client systems.

Although the preferred embodiment(s) have been described in relation to client systems having no mass storage device, it will be appreciated that the present invention is also applicable to client systems having some non-volatile storage such as a hard file or similar for the local storage of software and/or data.

In addition, in the preferred embodiment, a single server system carries out the monitoring and prediction functions and also transfers the predicted module to the client. In an alternative configuration, the monitoring and prediction steps could be undertaken at one server and the storage and transfer of software modules be undertaken at a second server.

CLAIMS

1. A method for pre-emptively transferring software modules in a data processing network from a host data processing system for execution on a target data processing system, wherein there are a plurality of types of software modules, each type providing a different service at the target system, the method comprising:

determining, at the host system, the type of a first software module transferred to a target system;

predicting, based on the determination of the type of the first module, a second software module type that may next be required at the target system;

automatically transferring a module of the second type to the target system.

2. A method as claimed in claim 1 comprising the further step of :

compiling, at the server, a history of the software module usage at each target system, the prediction step being based on the compiled history for each target system.

3. A method as claimed in claim 1 or claim 2 comprising the further step of monitoring the level of traffic on the network and automatically transferring the second software module type only when the traffic is below a threshold value.

4. A method as claimed in any preceding claim wherein the first software module is transferred from the host system in response to a request from the target system.

5. A method as claimed in any preceding claim wherein the software module is a dynamically loadable software module.

6. A method as claimed in claim 5, wherein the dynamically-loadable software module is a Java applet for execution via a java-enabled browser or java virtual machine on the target system.

7. A method as claimed in any preceding claim, wherein the network is a local area network.

8. A method as claimed in any of claims 1 to 6 wherein the network is an internet or intranet.

9. A method as claimed in any preceding claim wherein the first software module provides a text editing service at the target system.

10. A method as claimed in claim 9 wherein the second software module provides an e-mail communication service at the target system.

11. A data processing network comprising a server data processing system connected for communication to a plurality of target data processing systems, the data processing network including:

means, at or associated with the server data processing system, for transferring a plurality of types of software modules to the plurality of target systems for execution thereon, each type of software module providing a different service at a target system;

means for monitoring the identity of the software modules sent to the target systems;

means, responsive to a determination that a first type of software module has been communicated to a target system, for predicting that a software module of a second type will be subsequently required at the target system, and for causing the transferring means to pre-emptively transfer a software module of the second type to the target system; and

means, at the target system, for receiving the software modules from the host system.

12. A data processing network as claimed in claim 11, further including:

data gathering means for compiling a history of software module usage at each of the target systems, wherein the predicting means bases the prediction on the history for each target system.

13. A data processing network as claimed in claim 11 or claim 12 further including:

means for monitoring the level of traffic on the network and automatically transferring the second software module type only when the traffic is below a threshold value.

14. A data processing network as claimed in any of claims 11 to 13 wherein each software module type is a dynamically loadable software module.

15. A data processing network as claimed in claim 14, wherein the dynamically-loadable software module is a Java applet, each target system including a java-enabled browser or java virtual machine.

16. A server data processing system comprising:

means for communicating with a plurality of client data processing systems;

means for transferring a plurality of types of software modules to the plurality of client systems for execution thereon, each type of software module providing a different service at a target system;

means for monitoring the identity of the software modules sent to the target systems; and

means, responsive to a determination that a first type of software module has been communicated to a target system, for predicting that a software module of a second type will be subsequently required at the target system, and for causing the transferring means to pre-emptively transfer a software module of the second type to the target system.

The Patent Office

13

Application No: GB 9724364.6
Claims searched: 1-16

Examiner: Mike Davis
Date of search: 19 May 1998

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.P): G4A (AFL)

Int Cl (Ed.6): G06F

Other: Online: WPI

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
	None	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

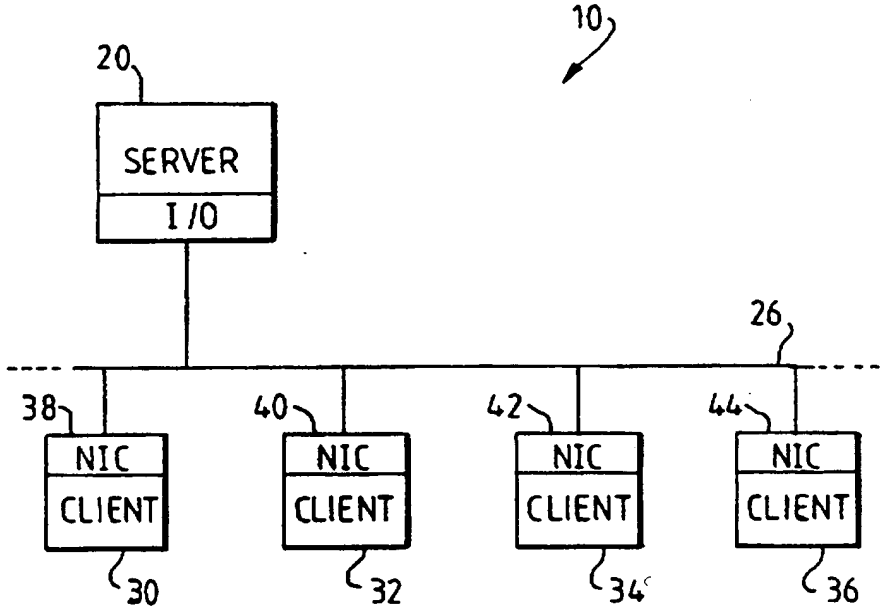
10
20
SERVER
I/O
26
38
40
42
44
NIC
CLIENT
30
32
34
36

FIG. 1

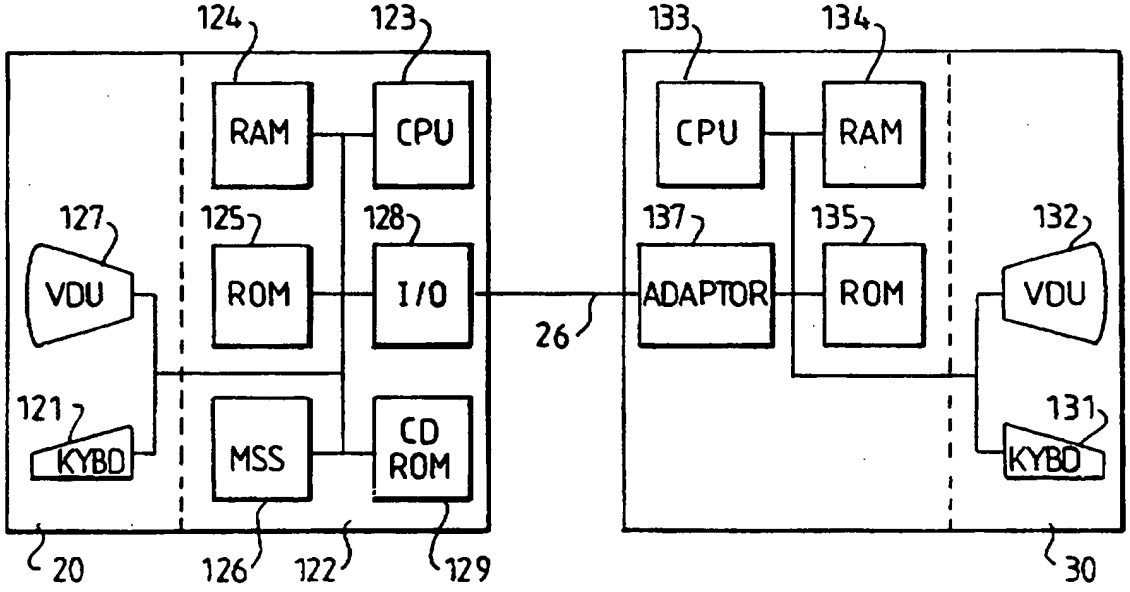
124
123
133
134
RAM
CPU
CPU
RAM
127
125
128
137
135
132
VDU
ROM
I/O
ADAPTOR
ROM
VDU
26
121
131
KYBD
MSS
CD ROM
KYBD
20
126
122
129
30

FIG. 2

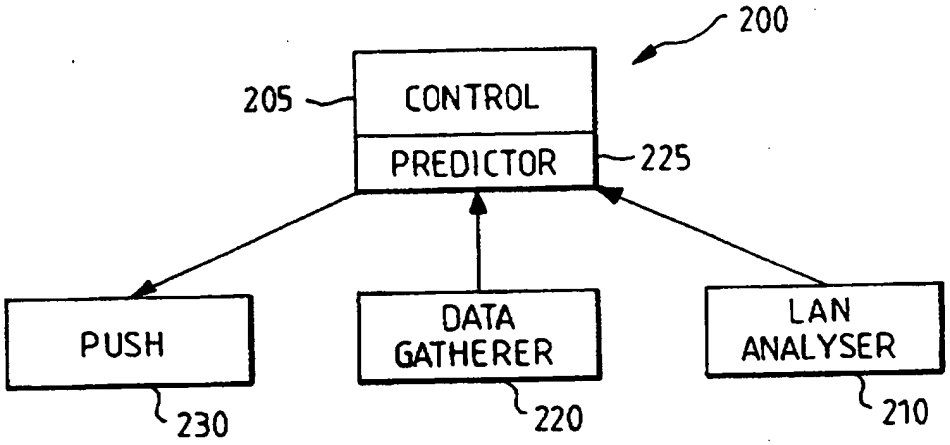

FIG. 3

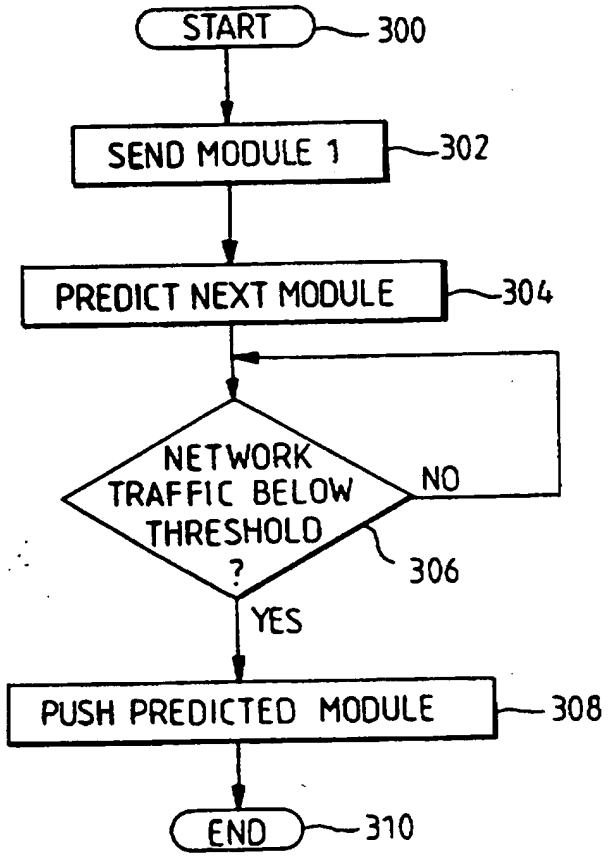

FIG. 4

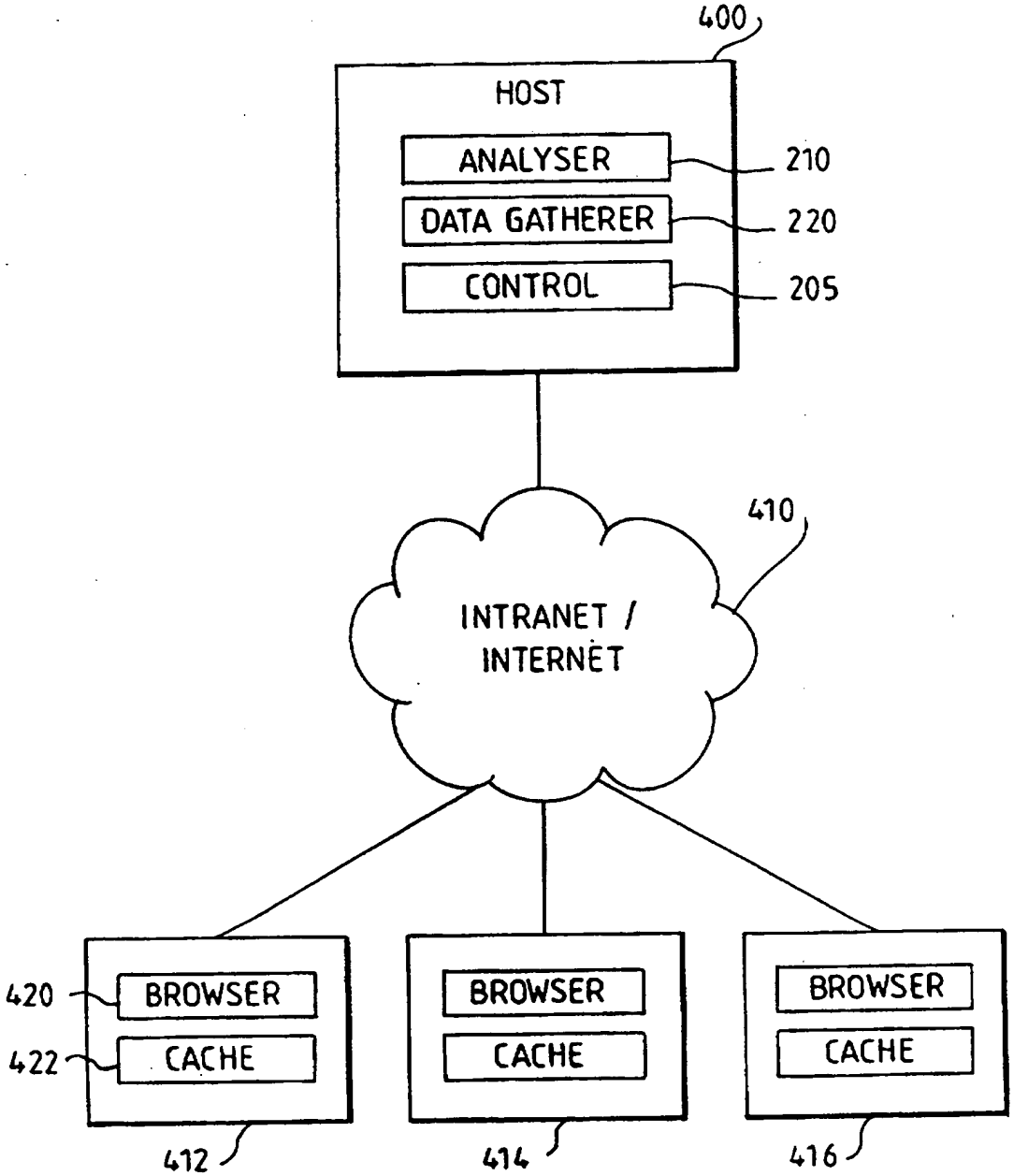

FIG. 5